**TCS INTERNSHIP REPORT**

**NAME OF THE INTERN:** RAMA JOSHITHA VURITI

**NAME OF THE PROJECT:** APPLICATION PACKAGING AND DEPLOYMENT

**NAME OF THE MENTOR :** RAHUL

**INTERNSHIP DURATION:** 45 DAYS

**INTERNSHIP END DATE:** 04-08-2023

**APPLICATION PACKAGING AND DEPLOYMENT**

Product packaging is a process that companies and organizations use to easily install required software on their customers' devices. It involves creating an information package for each brand or software used by the company. The benefits of this approach are enormous and lead to cost savings and increased productivity for the organization. It enables IT departments to have easier and faster access to all forms of software, including the latest models. Companies that want to be relevant and keep track of software and applications now need to adjust their strategies.

When searching for an application, the application's data is defined and its functions are defined. This is an important step because it forms the basis for the final design of the application package. The next step is to create the package. You'll create the necessary rules during exploration, so step two is usually easy. And the final step is UAT. This step is used to verify that the package is working as expected before sending it to production. In this step, you make the application behave as described in the discovery section.

Application packages, including updates, patches, and uninstallation, should be managed throughout their lifecycle. Management includes:

* Monitoring the package for any issues or errors.
* Applying updates and patches as needed.
* Uninstalling the package when it is no longer needed.

**WEEK 1**

**INTRODUCTION OF DOCKER**

Docker is a software platform that allows you to build, test, and deploy applications quickly. Docker packages software into standardized units called containers that have everything the software needs to run including libraries, system tools, code, and runtime. Using Docker, you can quickly deploy and scale applications into any environment and know your code will run.

Docker works by providing a standard way to run your code. Docker is an operating system for containers. Similar to how a Virtual Machine virtualizes (removes the need to directly manage) server hardware, containers virtualize the operating system of a server. Docker is installed on each server and provides simple commands you can use to build, start, or stop containers.

**DOCKER CONTAINER COMMANDS**

1.docker container create "name"

(To create a docker container)

2.docker container start "name"

(To start a specific docker container)

3.docker container ls -a

(To get the list of docker containers in your system)

4.docker container stop "name"

(To stop a specific docker container)

5.docker container restart "name" (or) docker container rm "name"

(To restart any specific docker container)

6.docker container ls -a -s

(For displaying the file size of the container)

7.docker container ls -a -l

(For showing the details of the latest container)

8.docker stats --format "table {{.Container}}\t{{.MemUsage}}\t{{.MemPerc}}"

(For displaying the memory usage of the container)

9.docker stats --format "table {{.Container}}\t{{.CPUPerc}}"

(For displaying the CPU usage of the container)

10.docker system df

(For displaying the disk usage)

**DOCKER COMMANDS** (for docker images)

1.cd/c & cd "copyfilepath"

(For setting the path)

2.docker image ls

(For getting the details of the docker image)

3.docker build -t "imagename" .

(For building a specific docker image)

4.docker run -p 9090:8080"imagename"

(For running the specific docker image

5.192.168.99.100:9090 & Localhost:8080

(For getting the output in the specific ports in the localhost)

**DOCKER COMPOSE COMMANDS**

1.docker-compose up

(For running the docker)

2.docker-compose start

(starts existing containers)

3.docker-compose stop

(stops the containers without removing them)

**WEEK 2**

**INTRODUCTION OF API’S AND CREATING TIMESTAMP REST API IN SPRINGBOOT**

The REST architecture builds web services that are accessible through URLs using one of four request verbs: POST, GET, PUT, and DELETE. So, you could say a REST API is software that allows you to create, read, update, and delete resources via URLs.

For this REST API, you will need a controller, a model, and a repository

@PostMapping: Allows you to create resources.

@GetMapping: Allows you to read resource data.

@PutMapping: Allows you to update resources.

@DeleteMapping: Allows you to delete resources.

Here we have used a Rest API for creating timestamp.That is explained as creating a Rest API to give the accurate time using spring boot.

We can send the request of the API by using POSTMAN by simply pasting the URL of the API with the localhost and the PORT which need to be used to show the result of the request.

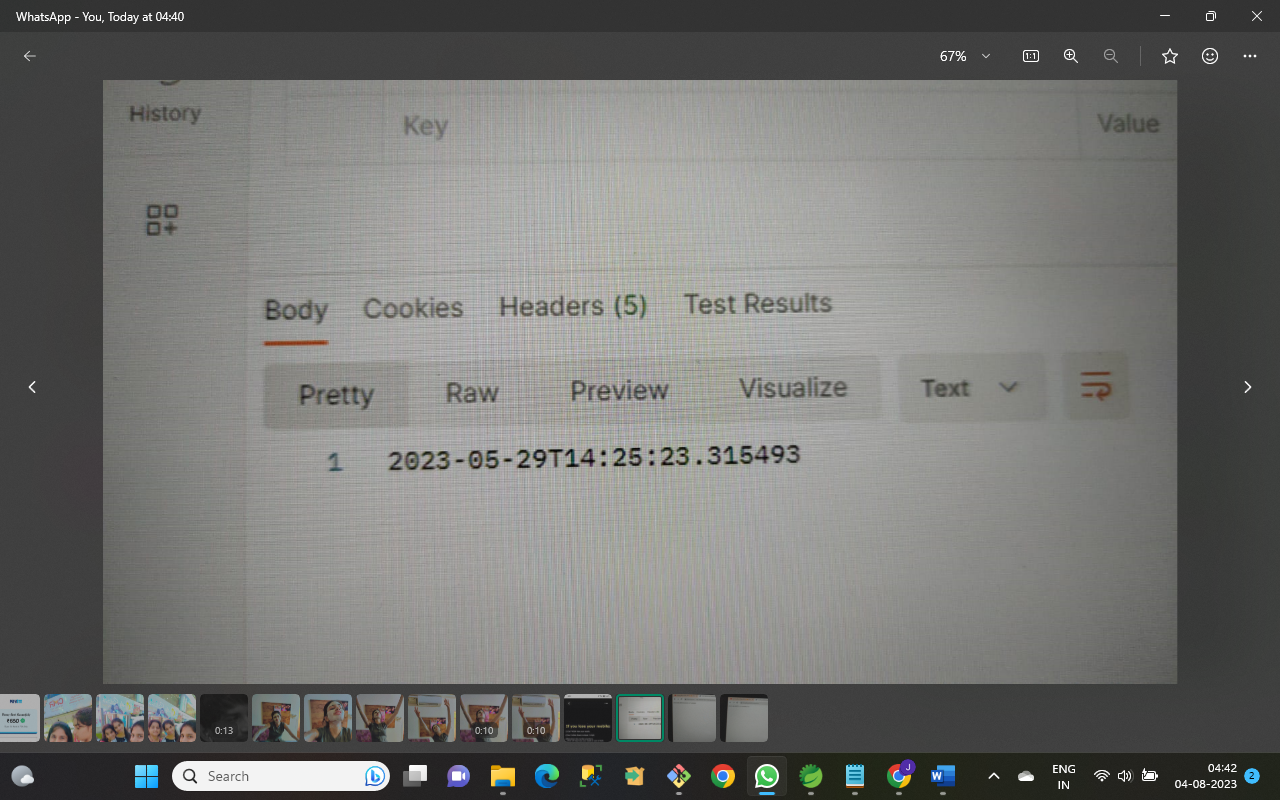


FIG: RESULT OF THE TIMESTAMP REQUEST IN POSTMAN

In the same way the request can be received in the localhost port also.

We can just type the localhost and the port number which we need to use with the URL like this : localhost:8080/datetime

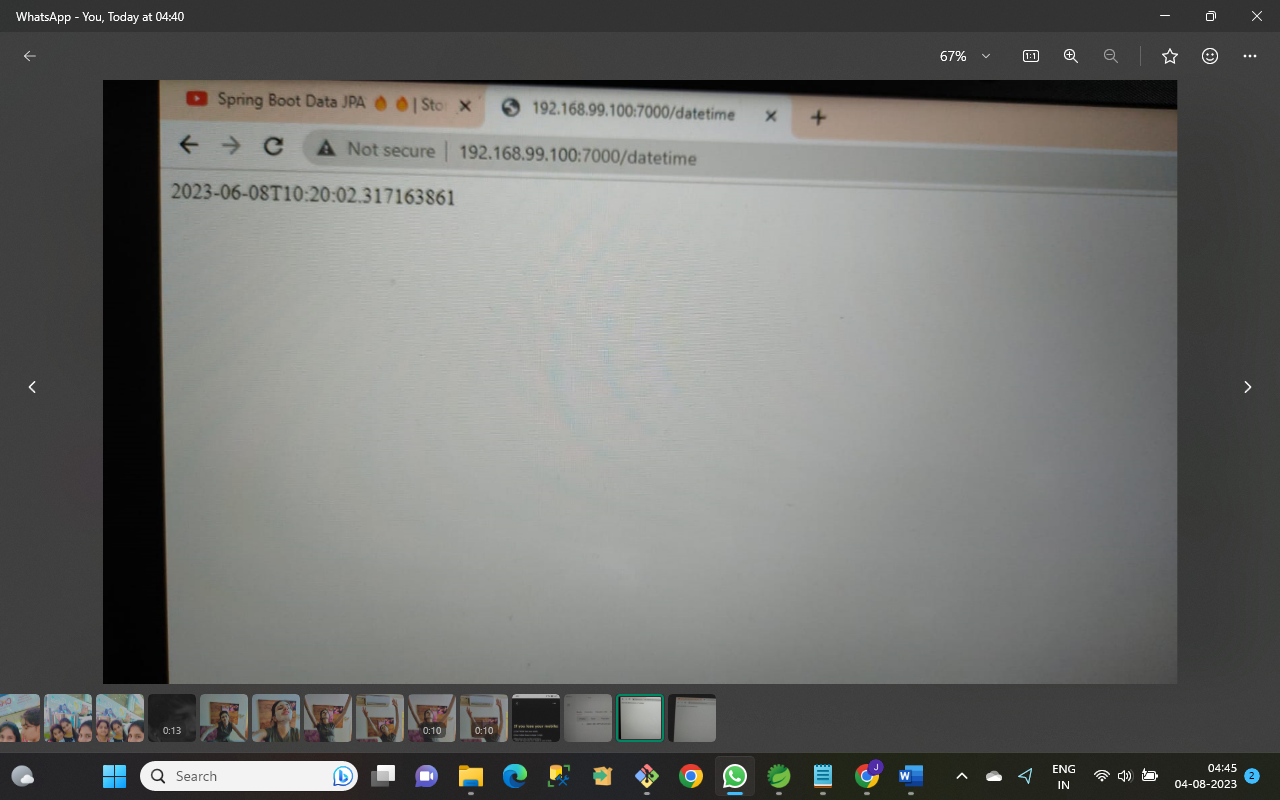


FIG : RESULT OF THE TIMESTAMP REQUEST IN LOCALHOST

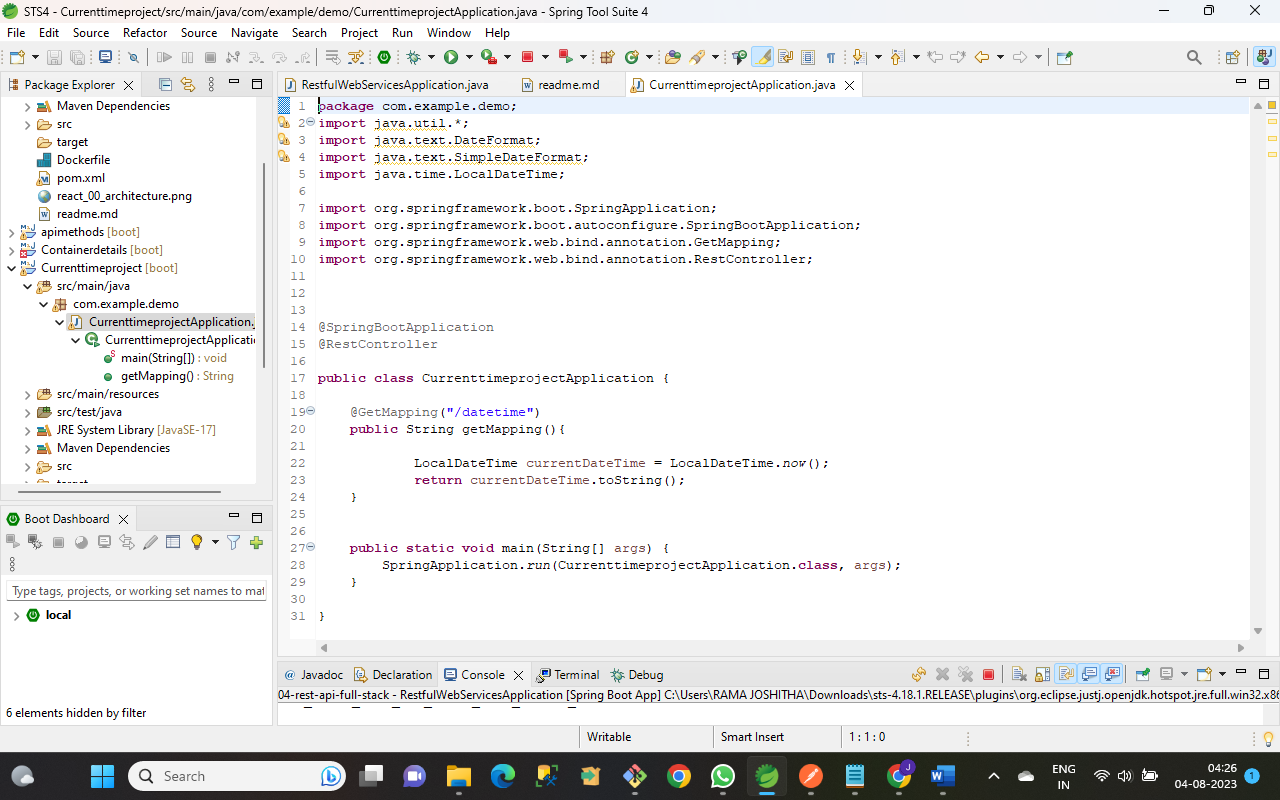


FIG : JAVASCRIPT USED IN SPRINGBOOT TO PRINT THE REQUEST OF THE TIMESTAMP

We should then containerize the REST API in Spring Boot using DOCKER.

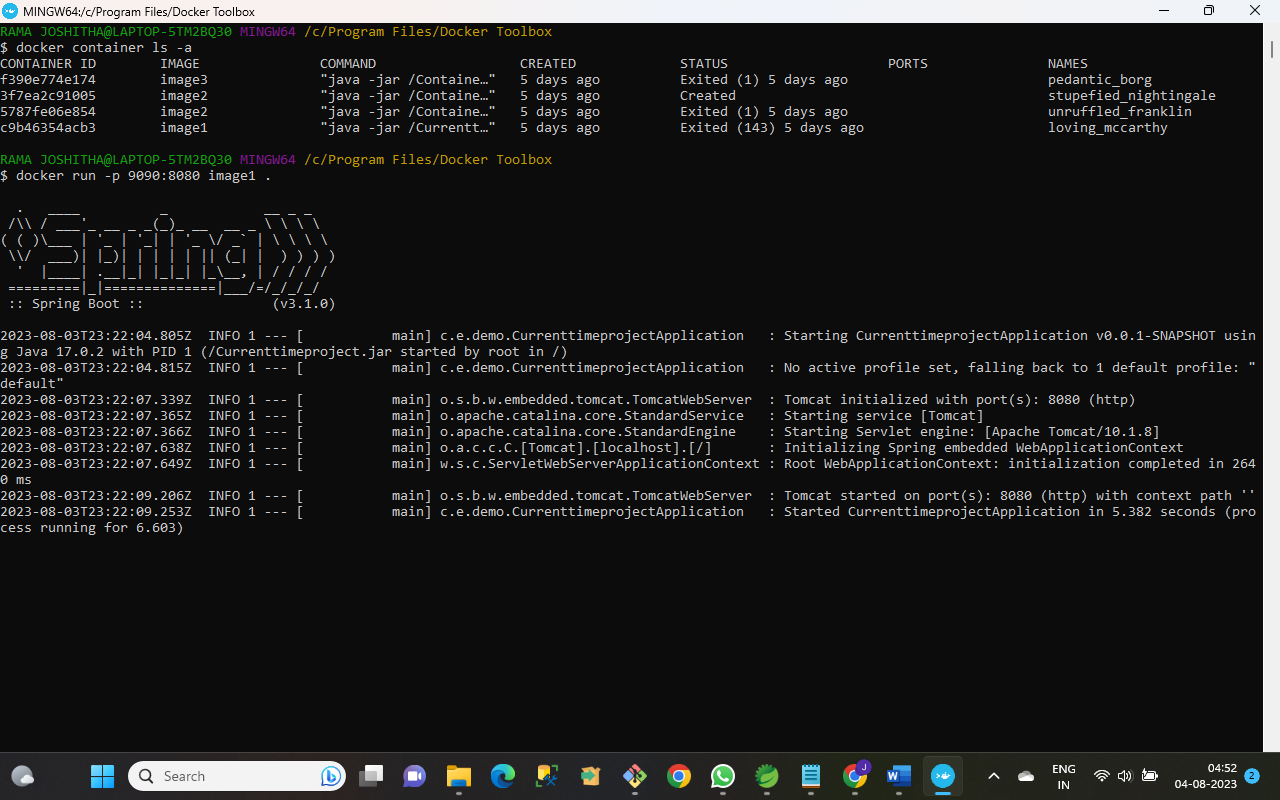


FIG : RUNNING THE API AFTER CONTAINERIZING WITH DOCKER

After containerizing with DOCKER , we can run it in the docker terminal.

Firstly we have to set the REST API path in the docker terminal.

Then we have to build the docker image

And then we have to run the docker commands in the docker terminal.

If both the Spring Boot application and the docker terminal is giving the same output , then your application is running successfully.

**WEEK 3**

**INTRODUCTION OF DOCKER COMPOSE**

Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application’s services. Then, with a single command, you create and start all the services from your configuration.

Compose works in all environments: production, staging, development, testing, as well as CI workflows. It also has commands for managing the whole lifecycle of your application:

* Start, stop, and rebuild services
* View the status of running services
* Stream the log output of running services
* Run a one-off command on a service

The key features of Compose that make it effective are:

* Have multiple isolated computers on a single host
* Preserves volume data when containers are created
* Only recreate containers that have changed
* Supports variables and moving a composition between environments

**TO-DO**

TO-DO is commonly used in task management and productivity contexts. It refers to a list of tasks or activities that need to be completed or accomplished within a specific period. People use to-do lists to help them stay organized, prioritize tasks, and ensure they don't forget important assignments or goals.

A to-do list typically contains various tasks, errands, or objectives, each represented as an item on the list. As you complete the tasks, you can check them off or mark them as done, providing a sense of accomplishment and progress.

To-do lists are popular tools for time management, productivity, and maintaining focus on essential tasks amid a potentially overwhelming workload. They can be used for personal or professional purposes and can be managed using pen and paper, digital apps, or other task management tools.

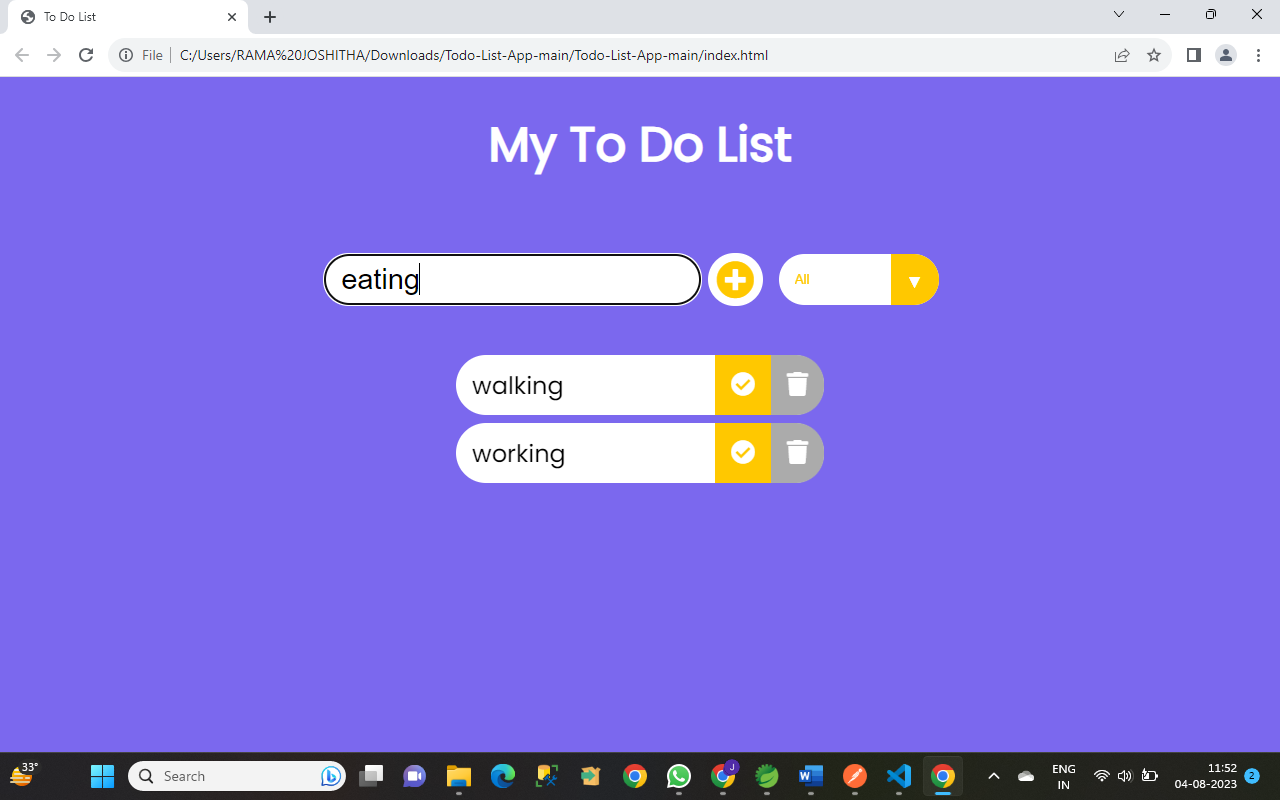


FIG:REPRESENTATION OF USING TO-DO

We can also use to-do by sending requests in the POSTMAN.

POSTMAN uses the URLS to GET , PUT , POST and DELETE the API requests.

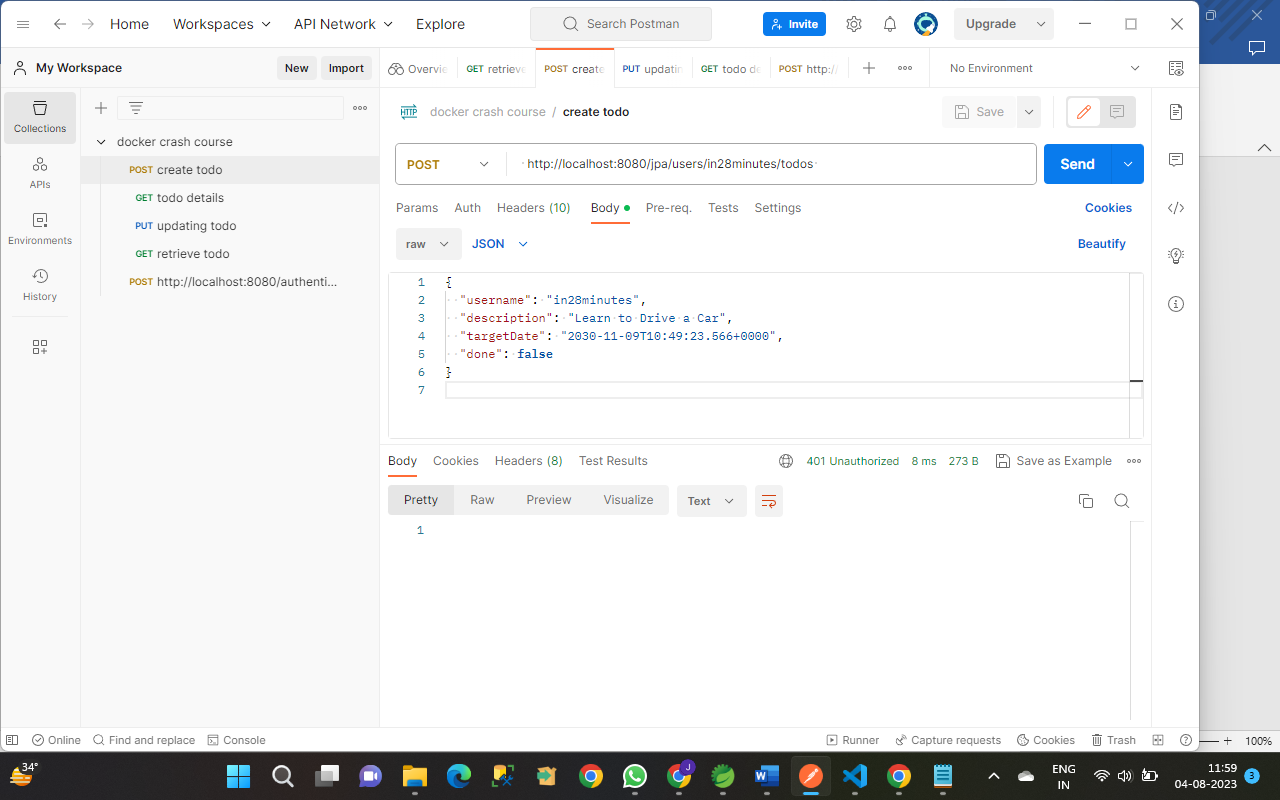


FIG: CREATING A TO-DO

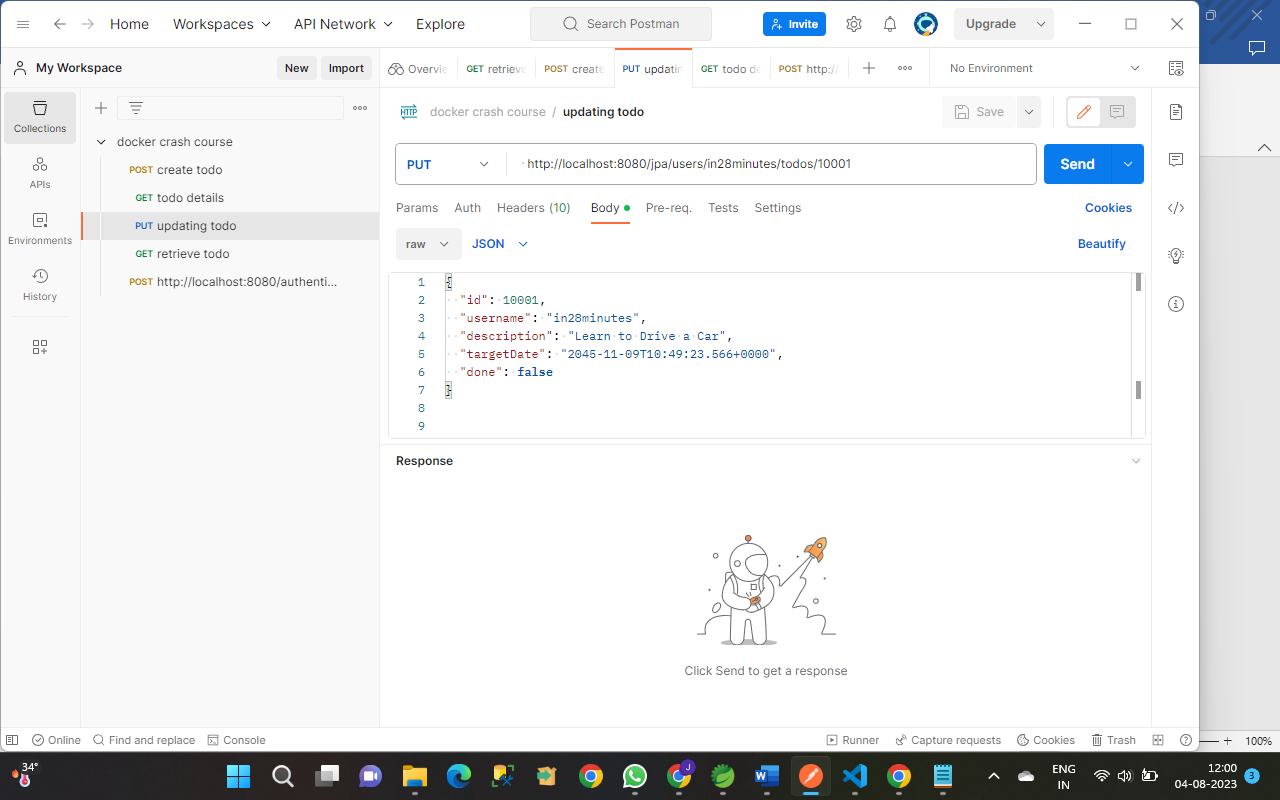


FIG: UPDATING A TO-DO

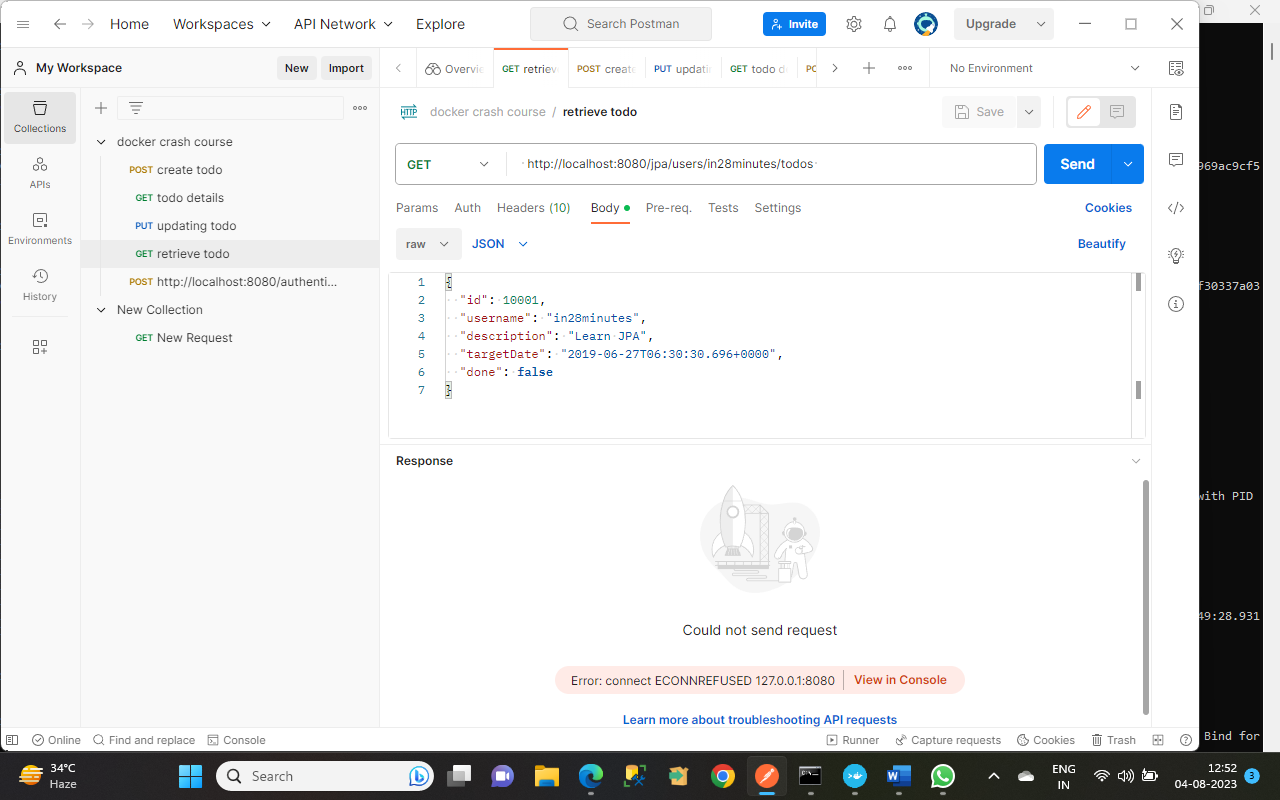


FIG: RETRIEVING A TO-DO

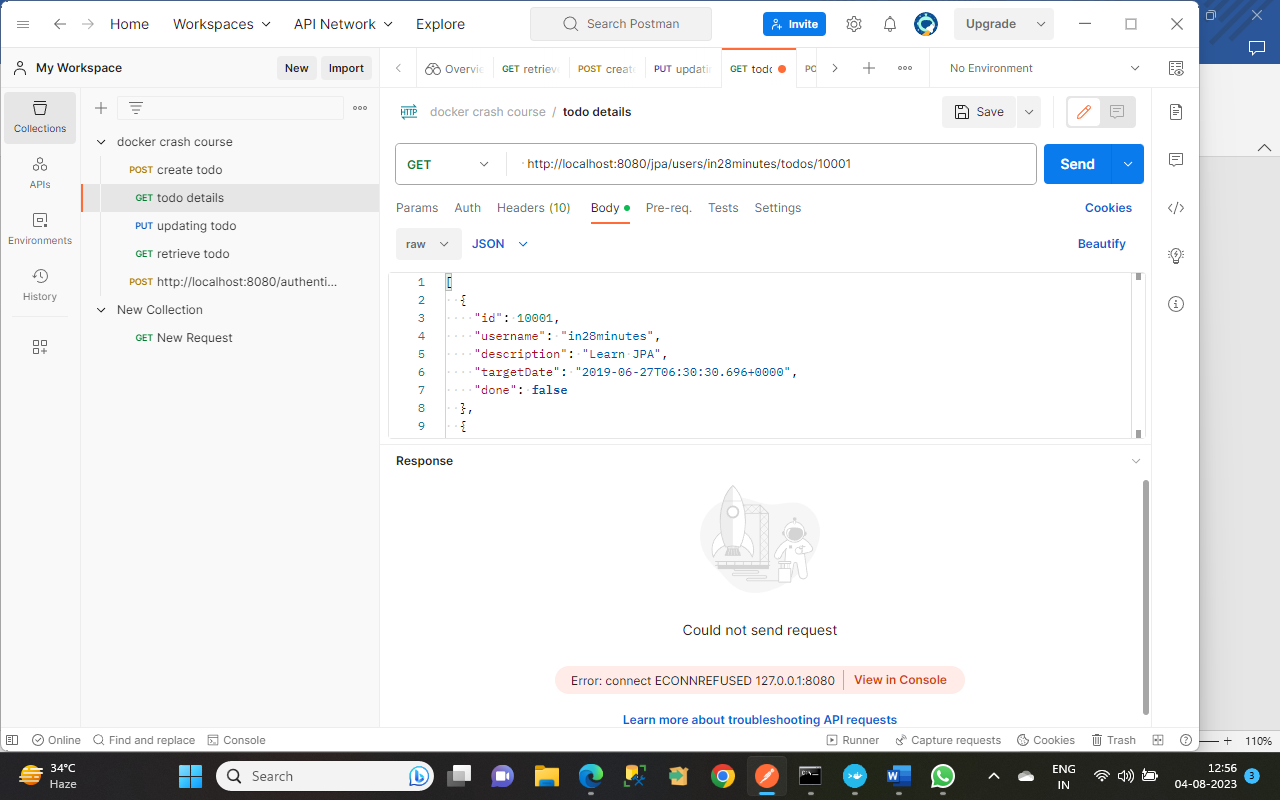


FIG: DETAILS OF A TO-DO

**WEEK 4**

**UI CREATION FOR DISPLAYING LIST OF DOCKER INSTANCES RUNNING ON SYSTEM**

To get the details of the instances of the docker running in the system, we use:

docker image ls

docker container ls -a

But to get the instances of docker running in a User Interface, we need front end and backend.

So here we used HTML, CSS, JAVASCRIPT for the front end and we used NODE.JS for the backend to display the instances in a UI

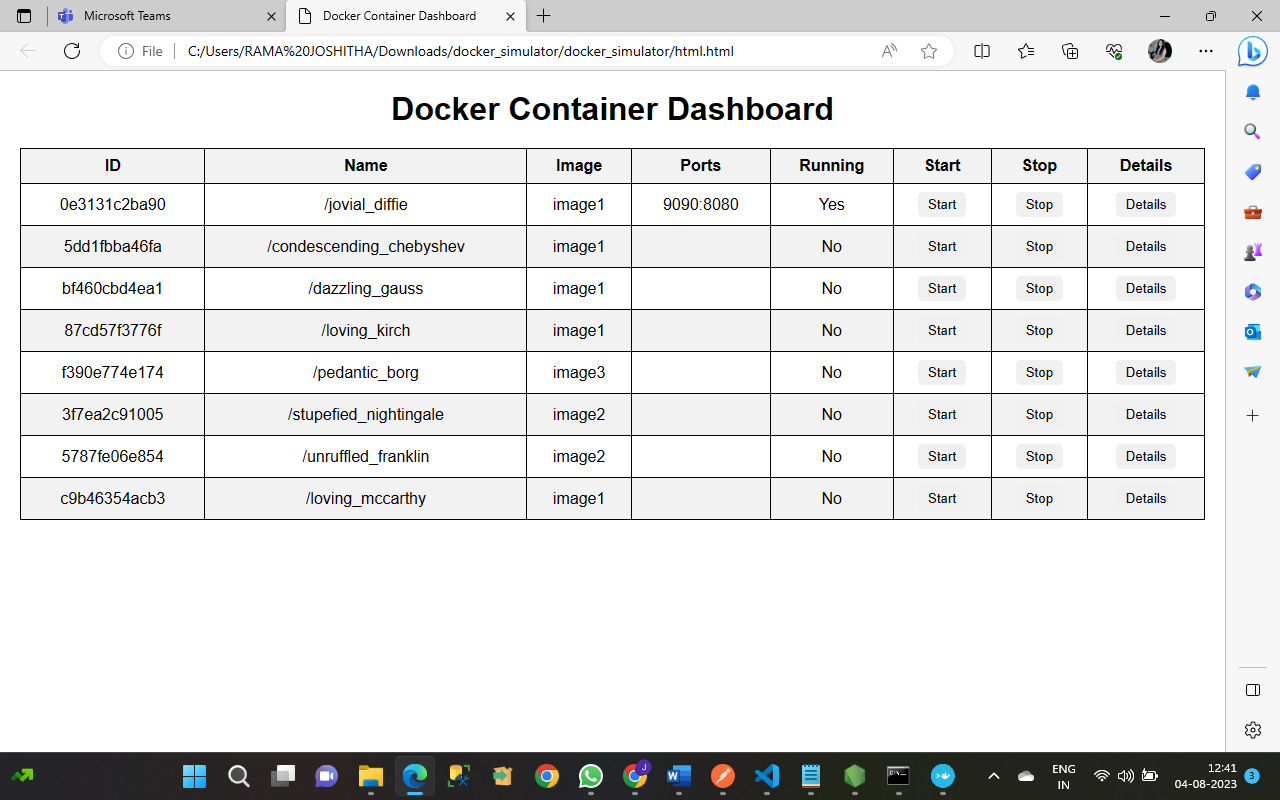


FIG: LIST OF DOCKER INSTANCES RUNNING IN A UI

Here we can see that the list of docker containers. If we start a specific container ,as we can see it starts running in the specified port number.

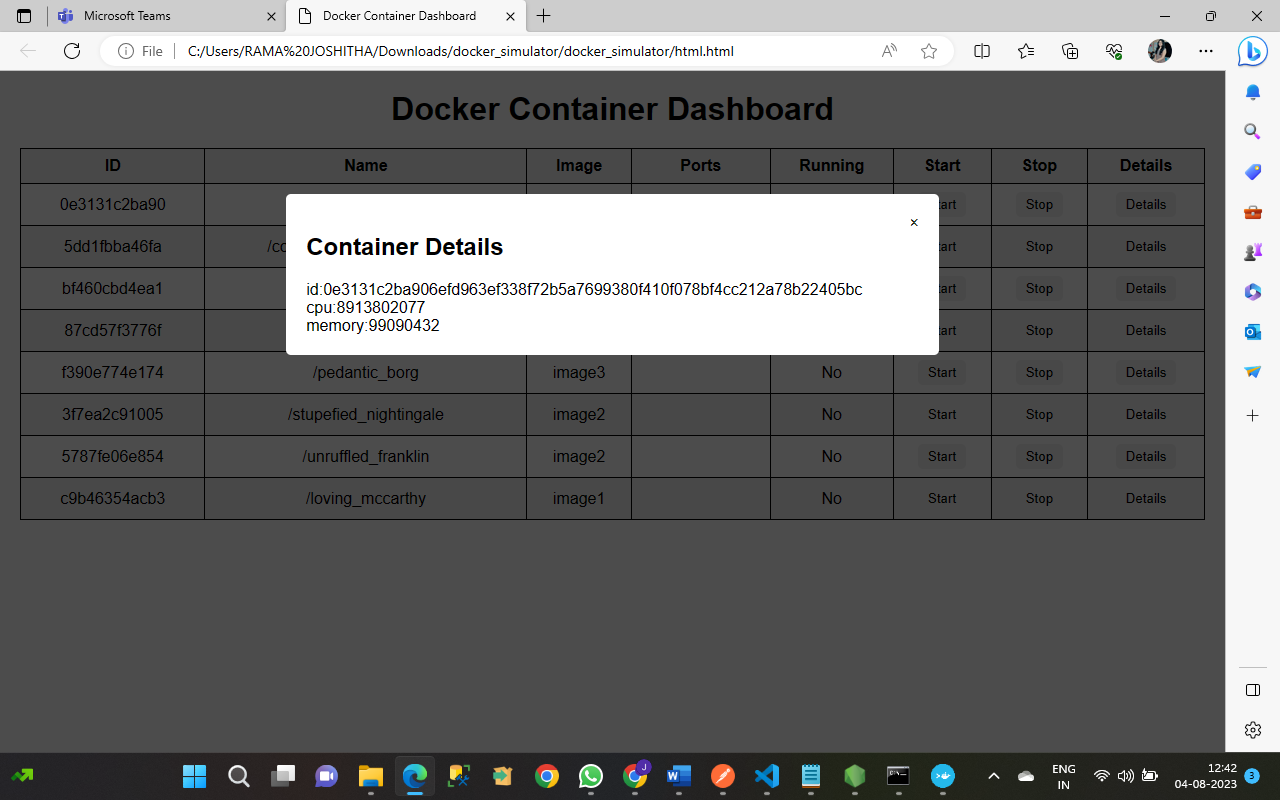
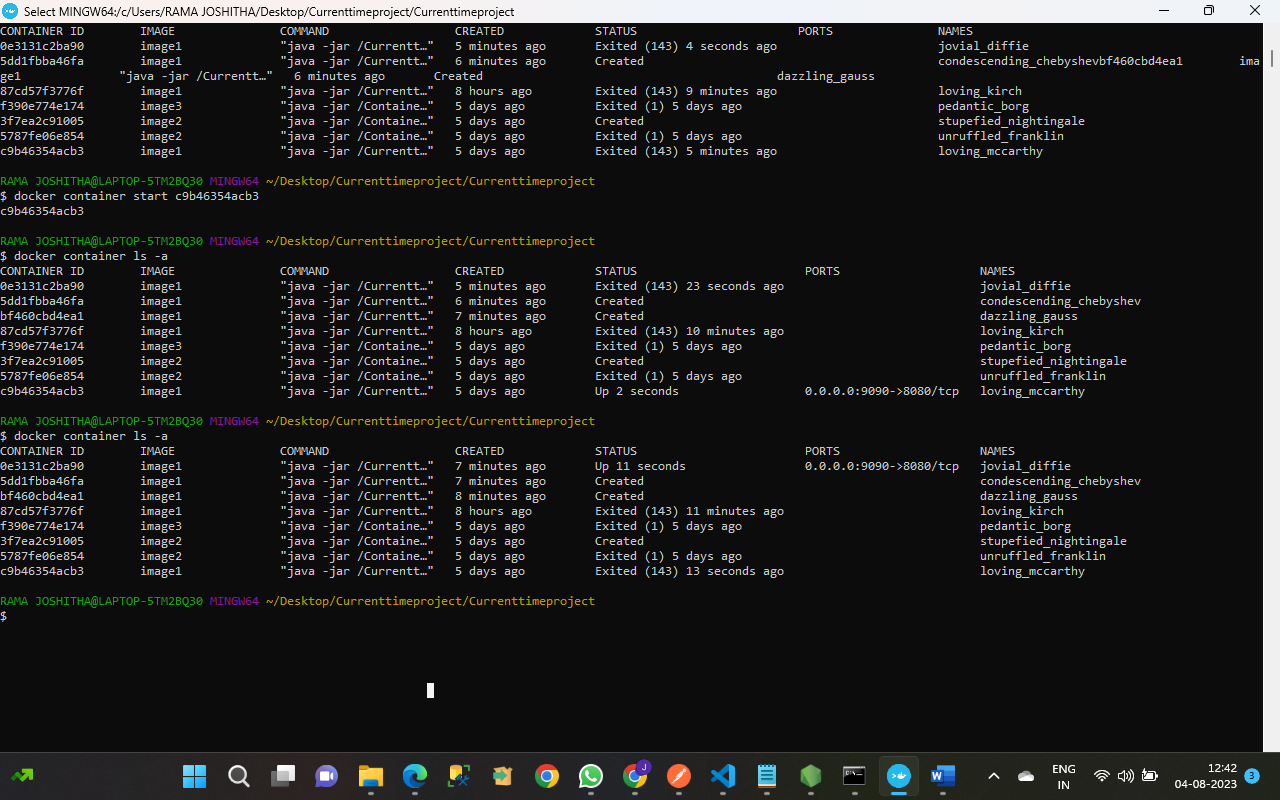


FIG: DETAILS OF THE CONTAINER RUNNING IN THE WEBPAGE.

We can also get the container details and its status by running the commands in the docker too.

FIG: DETAILS OF THE DOCKER CONTAINERS RUNNING IN THE DOCKER TERMINAL

So , in this Internship , I have learned of how various APIs work and deploy and how should we dockerize it with the containers, how to get the instances of the docker in the command prompt , Docker Terminal and finally on the User interface by displaying the docker details on the dashboard. This makes the conclusion of our Internship Report.

I am really very thankful to the TATA CONSULTANCY SERVICES for giving me this wonderful opportunity to participate in this Internship . I would like to thank Rahul sir for guiding me in this entire journey of my internship.

**THANK YOU**

Yours Sincerely

RAMA JOSHITHA VURITI